ISyE 6416 – Computational Statistics - Spring 2016
Final Report

Team Member Names: Danielle Boccelli and William Henry Taslim

Project Title: Comparison of Classification Methods in Identifying Handwritten Digits

Problem Statement

The availability of big data, robust algorithms, and powerful computers has led to rapid progress in handwriting recognition technology. This technology plays a significant role in the technological advancement of recognizing the words on scanned documents and potentially, in effectively verifying signatures. Our team is interested in comparing the efficiency and effectiveness of various classification methods that we learned in ISyE 6416 in identifying handwritten digits. Methods examined include: the expectation–maximization (EM) algorithm, the random forest algorithm, and k-means clustering. The handwritten digits dataset used for this project was developed specifically for use in machine learning research.

Data Source

The data source for this project is the Semeion Handwritten Digit Dataset (available at: archive.ics.uci.edu/ml/datasets/Semeion+Handwritten+Digit). The dataset was originally collected for the Semeion Research Center of Sciences and Communication in Rome, Italy for usage in machine learning research - specifically, in classification.

The Semeion dataset consists of 1593 handwritten digits written by a total of 80 individuals. These digits were scanned, and transformed into a 16x16 "pixel" grayscale image (for a total of 256 data points for each handwritten digits). Each pixel is the average gray value from the digit, generated by stretching the image. The grayscale images were then converted to black and white images (boolean values), by using a threshold for what is considered black, and what is considered white. Each individual from which the data was collected, was asked to write each digit (zero through nine) twice: the first in their normal way of writing, and the second quickly (or sloppily compared to their standard writing). An example of the final image, after transformations, can be seen in Figure 1 on the next page.

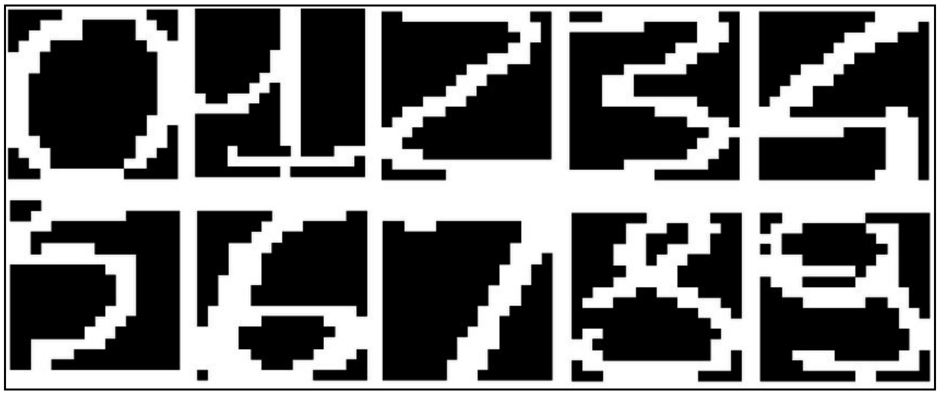

Figure 1: An example of the handwritten digits from the dataset

Methodology

The goal of the project is to compare different classification methods using the Semeion dataset. Although each individual is bound to have variations in their handwriting, the general structure of the individual digits should be enough to provide some distinction for the purposes of classification. Both k-means and the EM algorithm were coded in R, as well as all measures of accuracy. Random forest was implemented using an existing R package (randomForest), but accuracy will be coded in R as it was for k-means and EM. All R code for this project can be seen in Appendix B.

The original dataset was made up of 1597 rows and 266 columns - 256 of which were predictors (pixels), and 10 of which were response columns (rather than coding the response as a 0-9 digit in one column, the response was made up of a series of binary columns - one of which contained a 1 for each row, and the rest were 0s). In order to reduce the number of columns in the dataset, principal component analysis was completed using a built-in function in R (prcomp). The first ten principal components were run through each of the algorithms. Ten was chosen because that is the point when the added value of the next principal component starts to level off. This can be seen in Figure 2 on the next page.

Each method was run to try to reduce the number of misclassification that occurred. K-means was ran multiple times, with random starts, in order to minimize the possible effect of local minima in distance measurements. The EM algorithm was run until the expected log-likelihood reached a level of convergence (essentially, when the cluster identifications stopped changing). EM gives a probability that a point is in a cluster, and so the cluster with the highest probability at convergence will be taken as the final cluster for a given digit. Random forest was ran using 10-fold cross validation (splitting the data into ten sections, rotating through which section was the test set, and using the remaining nine sections as the training set at each iteration). The predictions for each testing set for random forest were combined into one full set of prediction values.

Accuracy was measured (for each method) by calculating misclassification rate (the percentage of digits that were not correctly identified). This was done by looking for the cluster that contained the highest number of occurrences of a digit, and dividing by the total number of occurrences of that digit. Care had to be taken to ensure that a cluster was not identified as being the main cluster for multiple digits (some digits have a lot of similarities, and may be almost evenly split between two clusters, which could make one cluster appear to be the real cluster for more than one digit). Total accuracy was measured by taking a weighted average of the misclassification rates.

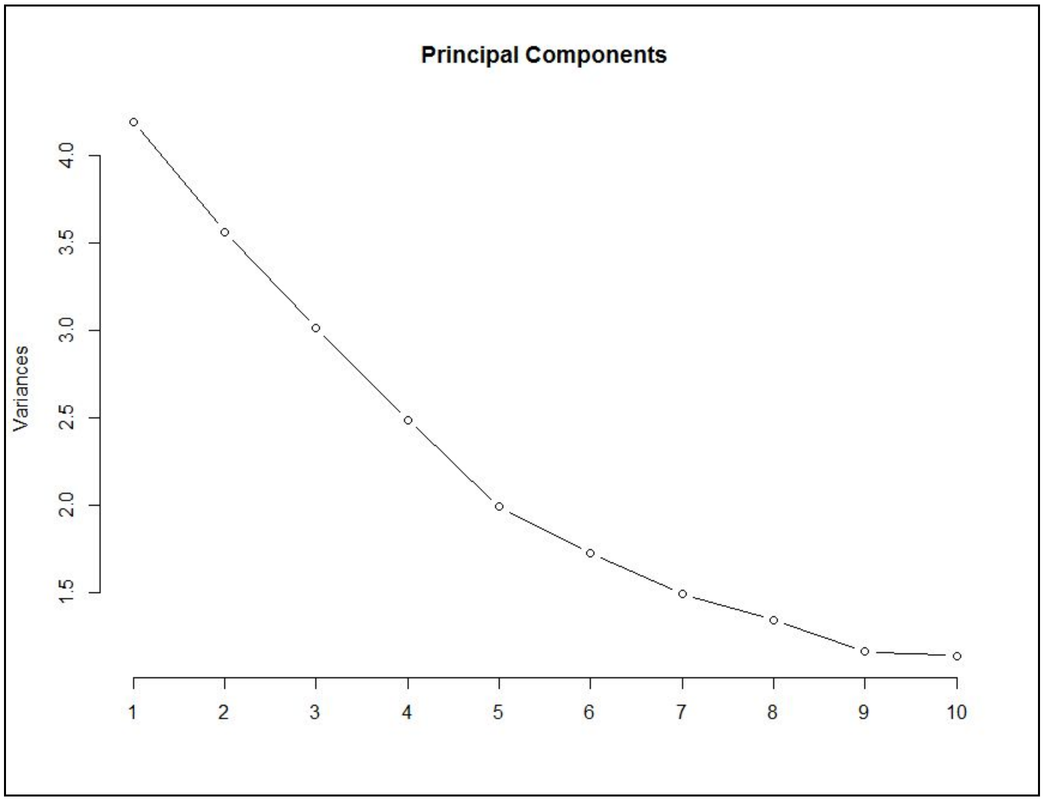

Figure 2: Principal Component Analysis Plot

Final Results and Evaluation

Using the methodologies listed previously, the three models were ran, and misclassifications rates were calculated. As can be seen in Figure 3 on the next page (and Table 1 in Appendix A), all models were able to properly classify more than half of the digits in the dataset (a random guess would only be accurate 10% of the time). Random forest was the most successful (with only 11.3%

misclassification), EM came in second (with 32.1% misclassification), and K-means was the weakest performer (with 41.5% misclassification).

There was also a difference in misclassification rates among the different digits. The digits 3, 6, and 7 were classified more accurately on average, and they were classified more accurately in the individual classification methods (when compared to the overall misclassification rate). The digits 1, 2, and 5 were classified with less accuracy overall, and they also underperformed the overall classification accuracy for each method.

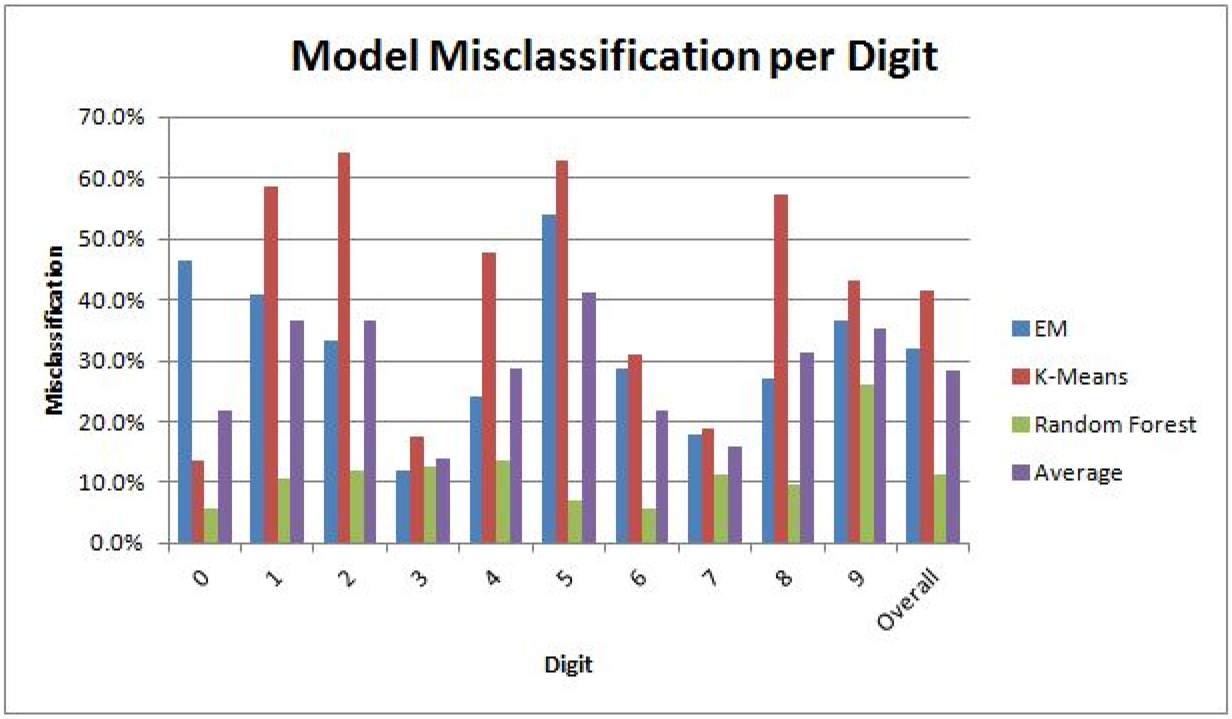

Figure 3: Model Comparison

Additional Analysis

Although the principal component analysis suggested that ten principal components be used, this was looked at in more depth using the most successful model - random forest. Figure 4 on the next page (and Table 2 in Appendix A), shows that there is a significant decrease misclassification from using 5 principal components, to using 10 principal components. Although misclassification decreases from 10 principal components to 15, and again when increase to 20, these changes are not as significant as the change from 5 to 10. The accuracy of some digits even increases when going from 10 to 15 (6, 8) and 15 to 20 (0, 4, 5, 8).

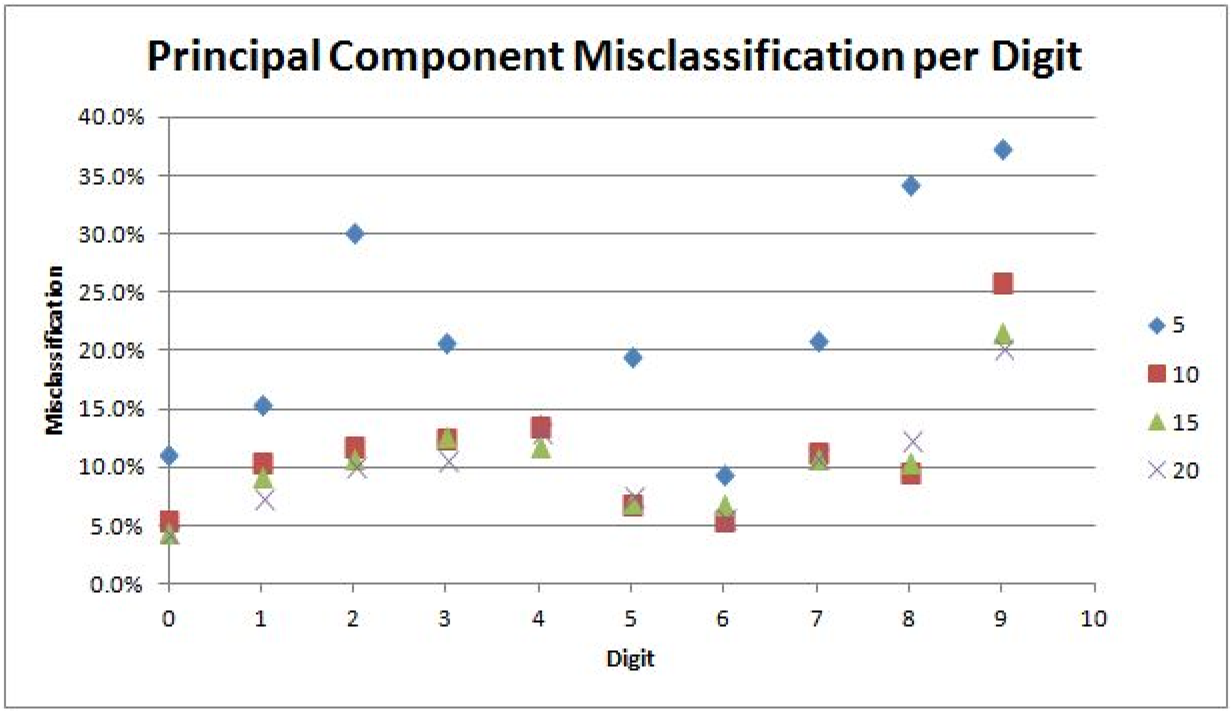

Figure 4: Principal Component Analysis Comparison

Appendix A

Table 1: Comparison of Methods and Digits				
	EM	**K-Means**	**Random Forest**	**Average**
0	46.6%	13.7%	5.6%	21.9%
1	40.7%	58.6%	10.5%	36.6%
2	33.3%	64.2%	12.0%	36.5%
3	11.9%	17.6%	12.6%	14.0%
4	24.2%	47.8%	13.7%	28.6%
5	54.1%	62.9%	6.9%	41.3%
6	28.6%	31.1%	5.6%	21.7%
7	17.7%	19.0%	11.2%	16.0%
8	27.1%	57.2%	9.7%	31.3%
9	36.7%	43.0%	26.0%	35.2%
Overall	32.1%	41.5%	11.3%	28.3%

Table 2: Principal Component Analysis				
	5	**10**	**15**	**20**
0	11.2%	5.6%	4.3%	4.4%
1	15.4%	10.5%	9.3%	7.4%
2	30.2%	12.0%	10.7%	10.1%
3	20.8%	12.6%	12.6%	10.7%
4	13.7%	13.7%	11.8%	13.0%
5	19.5%	6.9%	6.9%	7.6%
6	9.3%	5.6%	6.8%	5.6%
7	20.9%	11.4%	10.8%	10.8%
8	34.2%	9.7%	10.3%	12.3%
9	37.3%	26.0%	21.5%	20.3%
Overall	21.2%	11.3%	10.5%	10.2%

Appendix B
#EM Algorithm

```
require(mvtnorm)
require(emdbook)
require(randomForest)
set.seed(123456)

#Read in data and separate labels
data=read.csv("/dboccelli3/vlab/desktop/semeion.csv",header=FALSE)
X=data.matrix(data[,1:d])
Labels=apply(data[,257:266],1,function(xx){return(which(xx=="1")-1)})
d = 256
k = 10
n = 1593
q = 10

#Assign clusters
Z = array(0,c(n,k))
kmeansModel = kmeans(X, centers=k, nstart=50,iter.max=30)
for (i in 1:length(kmeansModel$cluster)){
  for (j in 1:k){Z[i,j]=ifelse(kmeansModel$cluster[i]==j,1,0)}}

#Parameter initialization
muk = array(0,c(k,d))
muk = kmeansModel$centers
Nk = colSums(Z)
pik = Nk/n
bigSigma = array(0,c(d,d,k))
V = array(0,c(d,d,k))
for(i in 1:k){sumV = matrix(0,d,d)
  for(j in 1:n){sumV = sumV+(X[j,]-muk[i,])%*%t(X[j,]-muk[i,])*Z[j,i]}
  V[,,i] = sumV/Nk[i]
  SVD = svd(V[,,i])  #from notes
  littleSigma = 1/(d-q)*sum(SVD$d[(q+1):d])
  Wq = SVD$v[,1:q]%*%diag(sqrt(SVD$d[1:q]-littleSigma))
  bigSigma[,,i] = Wq%*%t(Wq)+littleSigma*diag(1,d,d)}
P = matrix(0,n,k)
for(i in 1:k){P[,i] = dmvnorm(X, muk[i,], bigSigma[,,i])}
likelihood = sum(log(P%*%pik))

for(step in 0:20){
  #Parameter update
  Z = matrix(0,n,k)
  for(i in 1:k){Z[,i] = dmvnorm(X, muk[i,], bigSigma[,,i])*pik[i]}
  Z = t(apply(Z,1,function(x){return(x/sum(x))}))
  Nk = colSums(Z)
  pik = Nk/n
  muk = (t(Z)%*%X)/Nk
  V = array(0,dim=c(d,d,k))
  for(i in 1:k){sumV = matrix(0,d,d)
  for(j in 1:n){sumV = sumV+(X[j,]-muk[i,])%*%t(X[j,]-muk[i,])*Z[j,i]}
  V[,,i] = sumV/Nk[i]
  SVD = svd(V[,,i])
  littleSigma = 1/(d-q)*sum(SVD$d[(q+1):d])
  Wq = SVD$v[,1:q]%*%diag(sqrt(SVD$d[1:q]-littleSigma))
  bigSigma[,,i] = Wq%*%t(Wq)+littleSigma*diag(1,d,d)}
  #Likelihood calculation
  for(i in 1:k){P[,i] = dmvnorm(X, muk[i,], bigSigma[,,i])*pik[i]}
  newLikelihood = sum(log(P%*%pik))
  likelihood = cbind(likelihood,newLikelihood)}

#New label calculation
newZ = array(0,c(n,k))
for(i in 1:n){newZ[i,which.max(Z[i,])] = 1}  #label = k with highest p
newLabels=apply(newZ,1,function(xx){return(which(xx=="1")-1)})

#5 Accuracy Assessment
misCatRateEM0=1-sort(table((newLabels[which(Labels==0)])),decreasing=TRUE)[
1]/length(which(Labels==0))
misCatRateEM1=1-sort(table((newLabels[which(Labels==1)])),decreasing=TRUE)[
1]/length(which(Labels==1))
misCatRateEM2=1-sort(table((newLabels[which(Labels==2)])),decreasing=TRUE)[
1]/length(which(Labels==2))
misCatRateEM3=1-sort(table((newLabels[which(Labels==3)])),decreasing=TRUE)[
1]/length(which(Labels==3))
misCatRateEM4=1-sort(table((newLabels[which(Labels==4)])),decreasing=TRUE)[
1]/length(which(Labels==4))
misCatRateEM5=1-sort(table((newLabels[which(Labels==5)])),decreasing=TRUE)[
1]/length(which(Labels==5))
misCatRateEM6=1-sort(table((newLabels[which(Labels==6)])),decreasing=TRUE)[
1]/length(which(Labels==6))
misCatRateEM7=1-sort(table((newLabels[which(Labels==7)])),decreasing=TRUE)[
1]/length(which(Labels==7))
misCatRateEM8=1-sort(table((newLabels[which(Labels==8)])),decreasing=TRUE)[
1]/length(which(Labels==8))
```

```
misCatRateEM9=1-sort(table((newLabels[which(Labels==9)])),decreasing=TRUE)[
1]/length(which(Labels==9))
overallMisCatEM=(misCatRateEM0*length(which(Labels==0))+misCatRateEM1*len
gth(which(Labels==1))+misCatRateEM2*length(which(Labels==2))+misCatRateEM
3*length(which(Labels==3))+misCatRateEM4*length(which(Labels==4))+misCatR
ateEM5*length(which(Labels==5))+misCatRateEM6*length(which(Labels==6))+mi
sCatRateEM7*length(which(Labels==7))+misCatRateEM8*length(which(Labels==8
))+misCatRateEM9*length(which(Labels==9)))/1597
```

#K means

```
predictorsK=data.matrix(data[,1:d])
responseK=apply(data[,257:266],1,function(xx){return(which(xx=="1")-1)})
prcompsK = prcomp(predictorsK)$x[,1:10]
#create random starts
centers = matrix(0,10,10)
for (i in 1:10){centers[i,]<-sample(c(0,1),10,replace=TRUE)}
centers = cbind(centers,1:10)

#PC Plot
plot(prcomp(predictorsK),type="l",main="Principal Components")

XNew = matrix(0,dim(prcompsK)[1],dim(prcompsK)[2]+1)
#assign random group
for(i in 1:dim(prcompsK)[1]){XNew[i,]<- centers[sample(1:10,1),]}

#calculate centroids
means = matrix(0,10,10)
for (k in 1:10){
  inCluster = matrix(0,dim(prcompsK)[1],10)
  for (i in 1:dim(prcompsK)[1]){if (XNew[i,11]==k){inCluster[i,]<-prcompsK[i,]}}
  means[k,]<- as.array(colMeans(inCluster[which(inCluster[,1]!=0),]))}

#minimize distance
cluster = matrix(0,dim(X)[1],1)
int=0
while (int<40){for (i in 1:dim(X)[1]){error = matrix(0,10,1)
    for(m in 1:10){error[m,] = sum((prcompsK[i,]-means[m,])**2)}
    min = which.min(error)
    cluster[i,]<-min}
  for (k in 1:10){inCluster = matrix(0,dim(X)[1],10)
    for (i in 1:dim(prcompsK)[1]){if (cluster[i,1]==k){inCluster[i,]<-prcompsK[i,]}}
    if (length(inCluster[which(inCluster[,1]!=0),])>10){
      means[k,]<- colMeans(inCluster[which(inCluster[,1]!=0),])}
    if (length(inCluster[which(inCluster[,1]!=0),])==10){
      means[k,] = inCluster[which(inCluster[,1]!=0),]}}
  int = int + 1}

misCatRateK0=1-sort(table((cluster[which(Labels==0)])),decreasing=TRUE)[1]/len
gth(which(Labels==0))
misCatRateK1=1-sort(table((cluster[which(Labels==1)])),decreasing=TRUE)[1]/len
gth(which(Labels==1))
misCatRateK2=1-sort(table((cluster[which(Labels==2)])),decreasing=TRUE)[1]/len
gth(which(Labels==2))
misCatRateK3=1-sort(table((cluster[which(Labels==3)])),decreasing=TRUE)[1]/len
gth(which(Labels==3))
misCatRateK4=1-sort(table((cluster[which(Labels==4)])),decreasing=TRUE)[1]/len
gth(which(Labels==4))
misCatRateK5=1-sort(table((cluster[which(Labels==5)])),decreasing=TRUE)[2]/len
gth(which(Labels==5))
misCatRateK6=1-sort(table((cluster[which(Labels==6)])),decreasing=TRUE)[1]/len
gth(which(Labels==6))
misCatRateK7=1-sort(table((cluster[which(Labels==7)])),decreasing=TRUE)[1]/len
gth(which(Labels==7))
misCatRateK8=1-sort(table((cluster[which(Labels==8)])),decreasing=TRUE)[1]/len
gth(which(Labels==8))
misCatRateK9=1-sort(table((cluster[which(Labels==9)])),decreasing=TRUE)[1]/len
gth(which(Labels==9))
overallMisCatK=(misCatRateK0*length(which(Labels==0))+misCatRateK1*length(
which(Labels==1))+misCatRateK2*length(which(Labels==2))+misCatRateK3*lengt
h(which(Labels==3))+misCatRateK4*length(which(Labels==4))+misCatRateK5*le
ngth(which(Labels==5))+misCatRateK6*length(which(Labels==6))+misCatRateK7
*length(which(Labels==7))+misCatRateK8*length(which(Labels==8))+misCatRate
K9*length(which(Labels==9)))/1597

#R Package Model
```
#Random Forest
```
predictors=data.matrix(data[1:1590,1:d])
response=apply(data[,257:266],1,function(xx){return(which(xx=="1")-1)})
set = rep(c(1,2,3,4,5,6,7,8,9,0),159)
prcomps = prcomp(predictors)$x[,1:10]
responses = c()
forestPrediction = array(0,dim=c(159,10))
for(i in 1:10){responses = c(responses,response[which(set==(i-1))])}
for (i in 1:10){testPredictors = prcomps[which(set == (i-1)),]
  testResponse = response[which(set == (i-1))]
  trainPredictors = prcomps[which(set != (i-1)),]
  trainResponse = response[which(set != (i-1))]
forestModel=randomForest(as.factor(trainResponse)~.,data=trainPredictors,
ntree=100)
forestPrediction[,i] = predict(forestModel,newdata=data.frame(testPredictors))}

forestPredictionFinal = c()
for (i in 1:10){forestPredictionFinal = c(forestPredictionFinal,forestPrediction[,i])}

misCatRateforest0=1-sort(table((forestPredictionFinal[which(responses==0)])),dec
reasing=TRUE)[1]/length(which(Labels==0))
misCatRateforest1=1-sort(table((forestPredictionFinal[which(responses==1)])),dec
reasing=TRUE)[1]/length(which(Labels==1))
misCatRateforest2=1-sort(table((forestPredictionFinal[which(responses==2)])),dec
reasing=TRUE)[1]/length(which(Labels==2))
misCatRateforest3=1-sort(table((forestPredictionFinal[which(responses==3)])),dec
reasing=TRUE)[1]/length(which(Labels==3))
misCatRateforest4=1-sort(table((forestPredictionFinal[which(responses==4)])),dec
reasing=TRUE)[1]/length(which(Labels==4))
misCatRateforest5=1-sort(table((forestPredictionFinal[which(responses==5)])),dec
reasing=TRUE)[1]/length(which(Labels==5))
misCatRateforest6=1-sort(table((forestPredictionFinal[which(responses==6)])),dec
reasing=TRUE)[1]/length(which(Labels==6))
misCatRateforest7=1-sort(table((forestPredictionFinal[which(responses==7)])),dec
reasing=TRUE)[1]/length(which(Labels==7))
misCatRateforest8=1-sort(table((forestPredictionFinal[which(responses==8)])),dec
reasing=TRUE)[1]/length(which(Labels==8))
misCatRateforest9=1-sort(table((forestPredictionFinal[which(responses==9)])),dec
reasing=TRUE)[1]/length(which(Labels==9))
overallMisCatforest=(misCatRateforest0*length(which(Labels==0))+misCatRatefor
est1*length(which(Labels==1))+misCatRateforest2*length(which(Labels==2))+mi
sCatRateforest3*length(which(Labels==3))+misCatRateforest4*length(which(Label
s==4))+misCatRateforest5*length(which(Labels==5))+misCatRateforest6*length(
which(Labels==6))+misCatRateforest7*length(which(Labels==7))+misCatRatefore
st8*length(which(Labels==8))+misCatRateforest9*length(which(Labels==9)))/159
7
```